ERROR CONCEALMENT FOR H.264 VIDEO TRANSMISSION

A Thesis
Presented to
The Academic Faculty

By

Camille Mazataud

In Partial Fulfillment
of the Requirements for the Degree
of Master of Science in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

August 2009

ERROR CONCEALMENT FOR H.264 VIDEO TRANSMISSION

Approved by:

Benny K. Bing, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Fred B-H. Juang
School of Electrical and Computer Engineering
Georgia Institute of Technology

Gee-Kung Chang
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: July 1, 2009

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CABAC	Context Adaptive Binary Arithmetic Coding
CAVLC	Context-Adaptive Variable-Length Coding
FMO	Flexible Macroblock Ordering
GOP	Group of Picture
IDR	Instantaneous Decoding Refresh
MB	Macroblock
MBA map	Macroblock Allocation map
MV	Motion Vector
PPS	Picture Parameter Set
QP	Quantization Parameter
SG	Slice Group
SPS	Sequence Parameter Set

SUMMARY

Video coding standards such as H.264 AVC (Advanced Video Coding) rely on predictive coding to achieve high compression efficiency. Predictive coding consists of predicting each frame using preceding frames. However, predictive coding incurs a cost when transmitting over unreliable networks: frames are no longer independent and the loss of data in one frame may affect future frames. In this thesis, we study the effectiveness of Flexible Macroblock Ordering (FMO) in mitigating the effect of errors on the decoded video and propose solutions to improve the error concealment on H.264 decoders.

After introducing the subject matter in Chapter 1, we present the H.264 profiles and briefly determine their intended applications in Chapter 2. Then we describe FMO and justify its usefulness for transmission over lossy networks. More precisely, we study the cost in terms of overheads and the improvements it offers in visual quality for damaged video frames. The unavailability of FMO in most H.264 profiles leads us to design a lossless FMO removal scheme, which allows the playback of FMO-encoded video on non FMO-compliant decoders. In Chapter 3, we describe the process of removing the FMO structure but also underline some limitations that prevent the application of the scheme. Finally, we assess the induced overheads and propose a model to predict these overheads when FMO Type 1 is employed.

In Chapter 4, we develop a new error concealment method to enhance video quality without relying on channel feedback. This method is shown to be superior to existing methods, including those from the JM reference software and can be applied to compensate for the limitations of the scheme proposed in Chapter 3. After introducing our new method, we evaluate its performance and compare it to some classical algorithms.

CHAPTER 1

INTRODUCTION

The growing demand for video content on bandwidth-constrained networks such as the Internet and wireless networks provides the impetus for the development of new standards with high coding efficiency. Such standards attempt to maintain an acceptable level of visual quality for an encoded video while reducing its size. The H.264 advanced video coding (AVC) standard is the latest video coding standard of the ITU-T Video Coding Expert Group and the ISO/IEC Moving Picture Experts Group. H.264 is designed to improve the compression efficiency compared to previous MPEG codecs and to provide a reduction in visual artifacts such as blockiness, color bands, etc. It allows greater flexibility and scalability, which makes it become widely adopted in consumer electronics as well as for narrowband and broadband network transport.

H.264 compression is largely based on the same principles as previous MPEG standards. However, H.264 allows greater flexibility. The encoded video is divided into a series of access units called coded video sequences. Each access unit results in one coded frame but can provide additional information such as redundant or supplemental enhancement information. The frame is divided into slices which are a set of macroblocks (MBs) in raster scan order. One MB consists of a block of 16x16 pixels. Higher flexibility is achieved at the frame level due to techniques such as flexible MB ordering (FMO) or variable slice length coding. At the MB level, further compression efficiency is provided by finer prediction with MB subdivision of up to sixteen 4x4 blocks and an interpolation accuracy of one ¼ pixel for motion vector (MV) compensation.

H.264 employs three main types of frames: I-frames, P-frames, and B-frames. An I-frame (intra-coded frame) is encoded using information only from itself, and does not depend on

information from any other frame. Intra-coded MBs are also allowed in P- and B- frames but P-frames and B-frames can both use motion compensation prediction from reference frames: each block is predicted by displacing areas of reference frames. Unlike P-frames, B-frames can reference later frames in playback order for prediction purposes. For all types of frames, residual data representing the difference between the predicted MB and the actual pixels in the source video is transformed using an integer transform, and the transform coefficients are quantized and encoded into the bitstream.

For transmission over packet-based networks, the compressed video is embedded into a set of packets. Once sufficient information is received, the video is decoded and displayed if necessary. In low-delay applications such as live video streaming over the Internet, unreliable transmission protocols are used. In this context, buffer overflow at intermediate nodes or long queuing delays may lead to packet losses. With low-delay constraints, it may not be desirable to rely on feedback to compensate for these losses. In addition, the higher compression efficiency results in a large amount of information condensed into very few bits, which makes the loss of a part of the coded bitstream unacceptable. To address these problems, error concealment methods are introduced at the decoder (Figure 1) to mitigate the effect of losses.

H.264 exhibits robustness in a loss-prone environment. It offers a new set of error resilience tools (e.g., FMO, data partitioning) which are aimed at limiting the effect of loss propagation and at improving the effectiveness of error concealment algorithms.

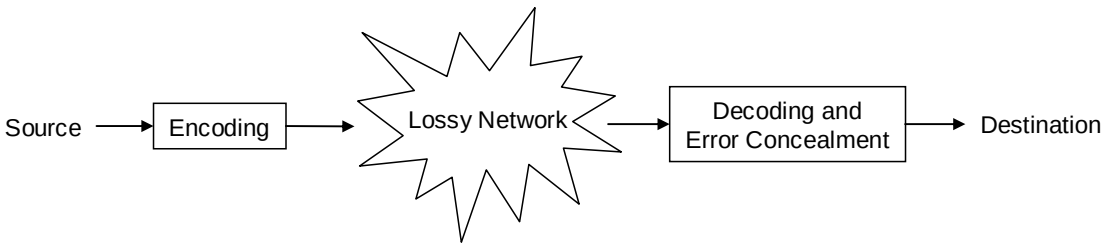


Figure 1 – Error concealment applied at the decoder.

CHAPTER 2

PROFILES AND FLEXIBLE MACROBLOCK ORDERING

The objective of the profiles in H.264 is to provide conformance points to facilitate interoperability between various applications of the standard. In section 2.1, we present the different profiles available in H.264. Then, we focus on FMO, which is an example of a feature not available in all profiles, in section 2.2.

2.1 Profiles in H.264

The scope of the H.264 standard is limited to the decoder by imposing restrictions on the bitstream and syntax, and defining the decoding process such as any decoder will produce similar outputs when decoding conforming bitstreams. Since H.264 provides adaptivity and scalability to address the needs of different applications when transmitted over heterogeneous networks, not all H.264 features are worth being used for a given application. For instance, decoders in error free environments are unlikely to integrate error resilience tools. Therefore, when encoding a video in a particular context only the features which may be understood by the targeted decoders should be used. In order to facilitate the interoperability between encoders and decoders [1] defines the profiles as “a subset of algorithmic features and limits that shall be supported by all decoders conforming to that profile”. In other words, the choice of a profile defines a set of coding tools which corresponds to the minimum capabilities of the decoder.

Table 1 - H.264 Profiles and Capabilities.

Parameters \ Profiles	Baseline	Main	Extended	High	High 10	High 4:2:2	High 4:4:4
Use of B-slices		X	X	X	X	X	X
Data Partitioning			X				
Interlaced Fields		X	X	X	X	X	X
Arbitrary Slice Ordering	X		X				
Multiple Slice Groups (FMO)	X		X				
8 Bit Sample Depth	X	X	X	X	X	X	X
9 Bit Sample Depth					X	X	X
10 Bit Sample Depth					X	X	X
11 to 14 Bit Sample Depth							X
Transform bypass Operation							X
Redundant Pictures	X		X				
Quantization Scaling Matrices				X	X	X	X
Weighted Prediction		X	X	X	X	X	X
CABAC		X		X	X	X	X
8x8 transform decoding				X	X	X	X
Separate Picture scaling				X	X	X	X
Separate Cb and Cr QP Control				X	X	X	X
Monochrome Format				X	X	X	X
Chroma Format 4:2:0	X	X	X	X	X	X	X
Chroma Format 4:2:2						X	X
Chroma Format 4:4:4							X

The choice of a profile at the encoding has no bearing on the video quality as long as all parameters in use are the same. However, if a different set of parameters is chosen a profile may provide additional capabilities for encoding, as summarized in Table 1.

Decoders conforming to the High 4:4:4 profile are capable of decoding a bitstream encoded with the High 4:2:2, High 10, High, and Main profiles. Similarly, High 4:2:2 profile decoders are capable of decoding the High 10, High, and Main profile bitstreams. These profiles do not offer tools for error robustness or resilience and are mainly designed for storage or for broadcasting in error free environments. They provide the capabilities for higher compression efficiency such as the use of weighted prediction for P-slices, 8x8 transform coding, etc. The higher profiles target more complex videos using more chroma samples per luminance sample (4:4:4 compared to 4:2:2 or 4:2:0) or finer quantization parameter values (up to 14 bits per sample for the High 4:4:4) and thus address higher quality videos.

The Baseline profile is a subset of the Extended profile. Both profiles address encoding in error-prone environments. The Extended profile offers more error resilience techniques (mainly data partitioning) and allows reducing temporal correlation (using B-frames) but also requires more computational power. The Baseline profile should be used when short encoding and decoding times are desired, i.e. with applications such as videoconferencing or live video streaming. For this reason, B-slices are not allowed in the Baseline profile whereas they are allowed in the Extended profile.

2.2 Flexible Macroblock Ordering

Predictive coding implies that frames are no longer independent. Thus, entire or partial frame loss may affect future frames thereby causing degradation to the entire sequence. H.264 offers a set of error resiliency tools aimed at limiting the effects of loss propagation.

2.2.1 Slice Coding and FMO

Slice coding prevents error spread at the frame level by segmenting the video frame into slices. A slice comprises an integer number of MBs coded in raster scan order when FMO is not in use. The slice structure of a frame is flexible and can be changed with each new frame. Slices are designed to be independently decodable meaning that a slice does not require other slices from the same frame to be decoded. Thereby, if a slice is lost or damaged the other slices of the frame may still be correctly received and thus can be properly decoded. Error concealment algorithms can then operate by using the information available from previously received frames or from correctly received slices.

In order to improve the concealment performances H.264 introduced FMO. FMO segments the frames in two to eight slice groups (SGs). A SG is a set of MBs which are not necessarily in raster scan order in the source frame. The frame is still divided into slices and each slice belongs to one unique SG and comprises MBs in raster scan order within this SG. One SG can include one or several slices. This process allows the assignment of MBs to slices in non-raster scan order at the frame level. For further details about FMO one can refer to [2].

When a slice is lost, FMO helps scattering the losses across the frame and thus allows error concealment to operate more effectively. For instance it is possible to choose a MB allocation map (MBA map) in such a way that the MBs of a slice are surrounded by MBs from other slices. If the first slice is lost then the reconstruction of the missing MBs is made much easier thanks to the existence of received neighbors. Thus, FMO improves the overall quality of the received video over lossy channels and has proved to be effective with loss rates of up to 10% for video-conferencing applications [3]. However, like many error resilience tools, FMO is only available in the Baseline and Extended H.264 profiles.

There exist seven types of FMO in H.264. Types 0 to 5 are presented in Figure 2. These Types contain a certain pattern which can be exploited to reduce the information needed for

reordering the MBs at the decoding. For FMO Type 0, each SG consists of a series of MBs in raster scan order: only the lengths of each SG need to be transmitted. FMO Type 1 uses a function known at the encoder and at the decoder which gives the location of the MBs depending on their position in the SG. FMO Type 2 is used to code separately one or more rectangular foreground SGs and a background SG: only the top left MB number and the bottom right MB number of each foreground SG are necessary for decoding. Types 3 to 5 use dynamic SGs which vary in size throughout the video by following predefined patterns. Type 6 is the most general and is not represented in Figure 2. It allows any MBA map and thus includes other types. However, for FMO Type 6 the MBA map must be explicitly coded into the bitstream.

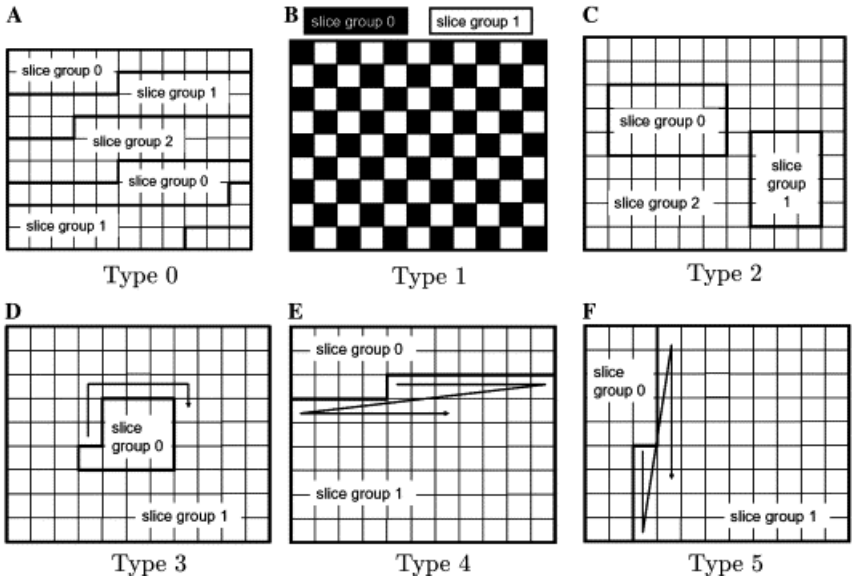


Figure 2 – Different FMO types.

In the remaining sections of this chapter, some FMO Types are tested (namely Type 1 with two and four SGs, Type 3, Type 4, and Type 5) under uniform packet loss condition for different slice lengths and group of picture (GOP) structures. Each packet contains one unique coded slice. We employ the Foreman CIF video (352x288 pixels per frame) with two different GOP structures (IPPPPPPPPPPP and IBBPBBPBBPBB abbreviated by IP and IBBP) and with quantization

parameter (QP) values of 20 or 30 for all frame types. For FMO Types 3, 4, and 5 the change rates for the number of MBs per SG is always chosen equal to the slice size: this allows a constant number of slices per picture to be kept. We consider slices of 33 MBs (12 slices per picture for all FMO Types), 66 MBs (6 slices for FMO Types 3, 4, 5, and 1 with two SGs and 8 slices for FMO Type 1 with four SGs), and 99 MBs (4 slices for all FMO Types). For the experiment, only I-, P-, and B-slices are removed (not the parameter sets). The slice-loss pattern is the same for all experiments – based on the uniform distribution. Loss percentages of 3%, 5%, and 10% are applied to each encoded video. The JM 12.4 reference software is used for encoding and decoding.

2.2.2 Overheads

The relative overheads when compared to a video encoded without FMO and with only one slice per picture are presented in Table 2. The sizes of the reference videos are presented in the "QP column".

Table 2 – Relative Overheads of FMO (%).

GOP	QP (Size of ref.)	MBs per Slice	Type 1 (2 SGs)	Type 1 (4 SGs)	Type 3	Type 4	Type 5
IP	20 (2484 kbytes)	33	12.2	15.9	4.4	4.6	4.3
		66	10.2	15.3	2.1	2.0	2.0
		99	9.4	14.7	1.3	1.3	1.1
	30 (587 kbytes)	33	34.7	41.7	15.4	16.1	15.2
		66	28.7	39.3	7.2	7.3	6.8
		99	26.7	36.9	4.5	4.7	4.0
IBBP	20 (2520 kbytes)	33	16.3	21.8	5.7	5.8	5.7
		66	13.3	21.3	2.7	2.7	2.5
		99	12.2	20.7	1.7	1.7	1.7
	30 (586 kbytes)	33	39.7	51.3	15.7	16.1	16.2
		66	31.7	49.0	7.5	7.4	7.2
		99	29.0	46.7	4.8	5.1	4.6

The file sizes vary according to the FMO Type but are always greater than the size of the file encoded without FMO. The relative overheads range from 1.1% for FMO Type 5 (with a QP value of 20, an IP GOP, and a slice size of 99 MBs) to 51.3% for FMO Type 1 (with four SGs with a QP value of 30, an IBBP GOP, and a slice size of 33 MBs). Besides, the cost of FMO Type 1 is considerably larger compared to other types (overheads are always at least two times larger for two SGs and three times larger with four SGs in most cases).

The first reason for this overhead is the reduction of the coding efficiency when several SGs, and thus slices, are used. This phenomenon can be observed when reading Table 2 column-wise: the overheads decrease when the slice sizes increase (i.e., the number of slices decreases). First, for intra-predicted MBs, neighboring MBs from other slices cannot be used to predict the textures so that the independence between slices is preserved. Next, the entropy coding methods applied in H.264 (CAVLC and CABAC) use context-based adaptivity to improve compression (see [4] for more details). Assuming that the motion is smooth locally, the correlation between the MVs of neighboring MBs is used to reduce the length of the syntax elements. The same idea is applied for residual information by assuming that the picture is smooth locally. In the case of several slices, the elements of neighbors from other slices cannot be used anymore and the coding efficiency worsens. This overhead is present for all FMO types because they require the creation of several slices. Nevertheless it is not comparable for all types. Although FMO Types 3, 4, 5, and Type 1 with two SGs have exactly the same number of slices in the tests, the differences observed between Type 1 with two SGs and the other Types range from 7.8% to 24.4%. The coding efficiency of FMO Type 1 is the worst. When two SGs are used, only the upper-right and the upper-left neighbors can be part of the same slice. With four SGs, the difference is even more pronounced because no neighboring MB is encoded in the same slice. Comparing FMO Type 1 with two and four SGs, we also observe that the difference between them is the largest when SGs of 66 MBs are used (31.7% and 49.0% with an IBBP GOP structure and QP values of 30). In this

situation, the use of four SGs results in the creation of three extra-slices and the coding efficiency worsens.

The second reason is the overhead due to extra information such as slice headers and extra picture parameter sets (PPSs). In this experiment, we only focus on the four FMO Types which do not require extra PPSs. However, even when no extra-PPS is needed, some extra bits are added to the already existing PPSs and to the slice headers. An example of the beginning of a PPS (actually the first 24 bits of 0xC46021210710) when FMO Type 5 is used is presented in Table 3. In this case, 3 bits indicate the number of slice groups, 6 bits indicate the FMO type, 1 bit indicates that the new MBs in the growing SG are added in raster scan order, and 11 bits indicate that 33 MBs must be added to the growing SG with each new instantaneous decoding refresh (IDR) picture. The end of the PPS does not carry information relative to FMO and would be the same if the bitstream were encoded without FMO thus it is not presented. This shows that all these overheads only account for a few bits and are minor compared to the size of a coded sequence.

Table 3 – Bit pattern of a PPS with FMO Type 5.

Bits	Meaning
1	pic_parameter_set_id = 0
1	seq_parameter_set_id = 0
0	entropy_coding_mode_flag = 0
0	pic_order_present_flag = 0
010	num_slice_groups_minus1 = 1
00110	slice_group_map_type = 5
0	slice_group_change_direction_flag = 0
00000100001	slice_group_change_rate_minus1 = 32

The GOP structure influences the relative overhead. All overheads are smaller when an IP structure is used compared to an IBBP structure. We can conclude that the use of B-frames allows a reduced temporal correlation at the expense of a higher relative overhead.

A last observation is that the relative overhead of any FMO Type is much smaller when the QP value decreases: the largest overhead for QP values of 20 is 21.8% compared to 51.3% for QP values of 30. In this situation, the phenomena depicted above still occur but the size of the video encoded without FMO is considerably larger (2,519,908 bytes for a QP value of 20 against 586,022 bytes for a QP value of 30 and an IBBP GOP) and the resulting relative overhead is smaller. Thus, we can conclude that the relative cost of FMO is more acceptable for higher quality videos.

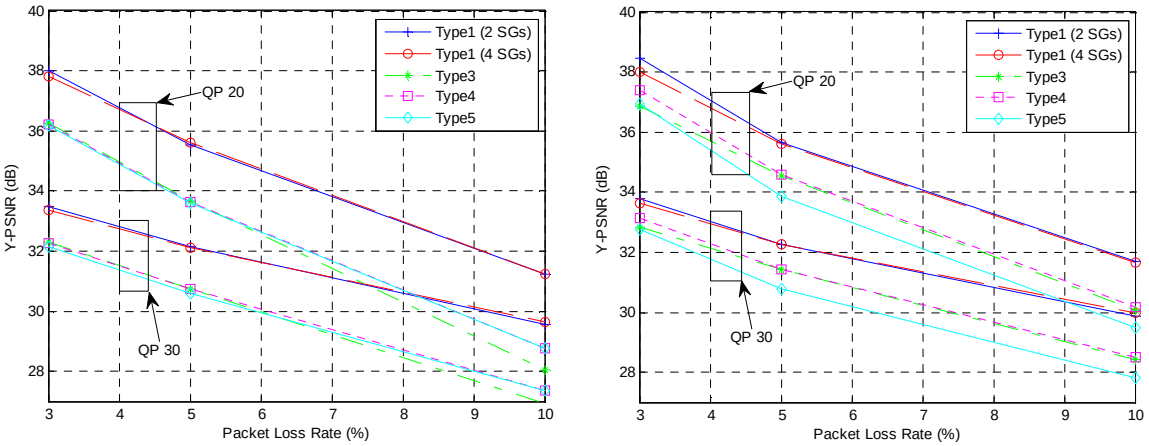
2.2.3 Error Resilience Performance

In order to evaluate the performances of the different FMO Types the Y-PSNR of the reconstructed videos is computed. The results are presented in Figure 3. In the context of this experiment and with the change rates chosen, the results obtained with FMO Type 4 are similar to those which would be gotten from a non FMO encoded video with the same number of slices (the slices are identical).

In all cases, FMO Type 1 (with two or four SGs) clearly outperforms other FMO types and the encoding without FMO. The difference is more pronounced when the losses increase and the QP values are lower.

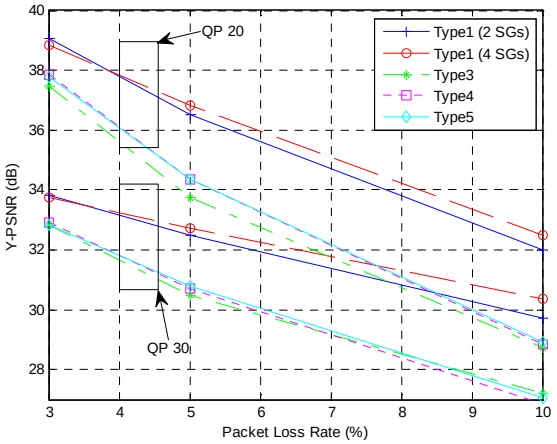
For small slices (33 MBs), regardless of the values of the other parameters, the IP GOP structure presents a higher average Y-PSNR than IBBP for all FMO Types studied. Nevertheless, for larger slices, no conclusions can be drawn concerning the GOP structure.

There is no correlation between the size of the slices and the average Y-PSNR either. However, the Y-PSNR only measures the distortion of each picture. A higher Y-PSNR does not guarantee better human perception (Figure 4). Subjective tests reveal that the videos with smaller slices are more pleasant to watch because the loss of a long slice degrades an important part of the picture whereas the loss of a small slice only damages a limited area.

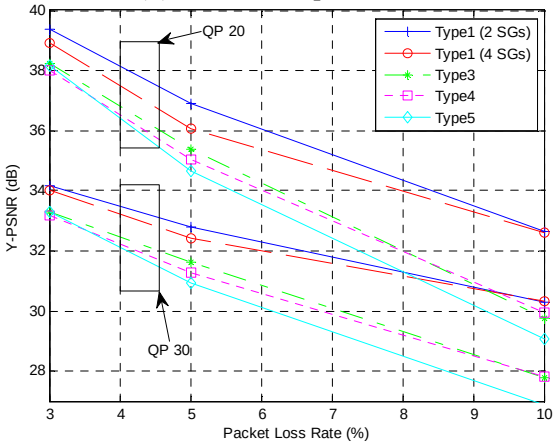


(a) IBBP 33 MBs per Slice

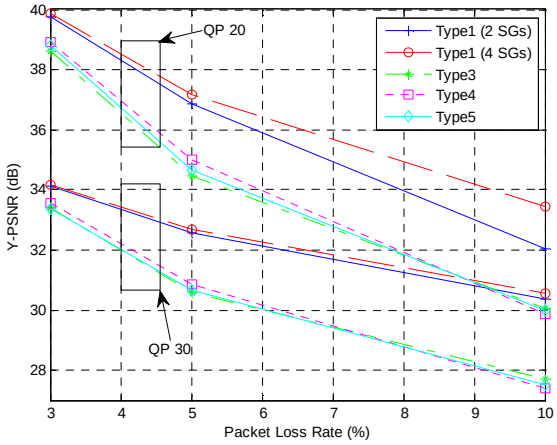
(b) IP 33 MBs per Slice



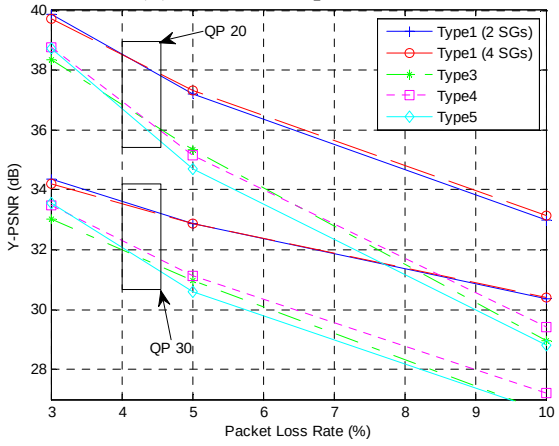
(c) IBBP 66 MBs per Slice



(d) IP 66 MBs per Slice



(e) IBBP 99 MBs per Slice



(f) IP 99 MBs per Slice

Figure 3 – Y-PSNR of concealed videos with FMO.

(a) Frame 33, Y-PSNR=30.99 dB.

(b) Frame 33, Y-PSNR=31.04 dB.

Figure 4 – Example of better Y-PSNR but worse subjective visual quality.

The performances obtained when varying the number of SGs for FMO Type 1 are almost the same on average. This means that, in the context of uniform random losses with loss probabilities lesser than 10%, it is not worth using more than two SGs. Indeed, increasing the number of SGs does not improve the error resilience but increases the cost of FMO as shown in the section 2.2.2.

The differences observed between FMO Types 3, 4, and 5 are random and there is no situation in these tests where one of the types outperforms the others. The differences observed are only due to the location of the lost slices in each picture, which differ from one type to the other. This is because none of these types scatters the MBs in better way than the others. Thus, the lost slices are more difficult to reconstruct compared to FMO Type 1 for which MBs surrounding lost MBs may be received.

CHAPTER 3

FMO REMOVAL

Like many error resilience tools, FMO is only available in the Baseline and Extended H.264 profiles. As a result many popular H.264 players cannot decode FMO-encoded videos. However, in section 2.2.3 we justified the usefulness of FMO when transmitting over lossy networks. This problem has been addressed at the *transmitter* for non FMO-compliant encoders in [5] and [6], where lossless transcoding techniques are employed to modify the slice structure and FMO is introduced separately after encoding the video content but before streaming.

In section 3.1 we introduce a lossless scheme at the *receiver* that allows the display of FMO-encoded videos by H.264 players that are non FMO-compliant (Figure 5). Unlike methods reported in [5,6] the scheme has the capability of modifying the slice structure when slices are lost during the transmission. We prove that the scheme is lossless in section 3.2. In section 3.3, we show how the method can also be used to reduce the size of videos encoded with several slices but without FMO. Then we assess the cost of the method via actual implementation and propose a model for predicting the overheads induced by the technique when FMO Type 1 is used in section 3.4.

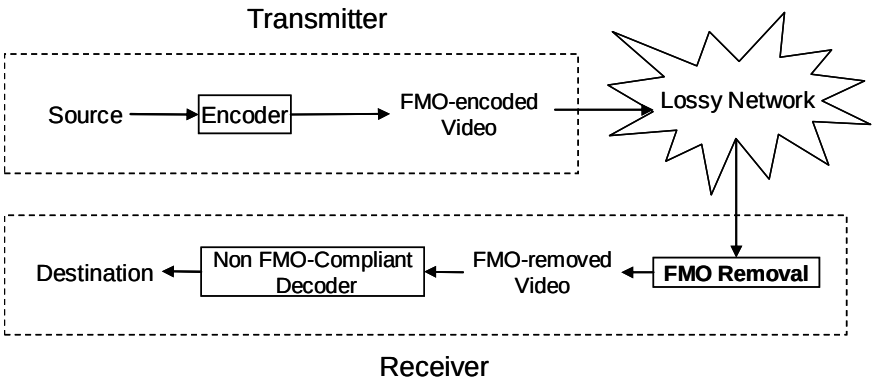


Figure 5 – The FMO removal scheme is applied at the receiver before decoding.

3.1 FMO Removal Scheme

We now describe the overall process of decoding a H.264-encoded video. First, the beginning of the bitstream is parsed to extract the information related to the entire stream, or at least to a sequence of frames between two IDR pictures coded into the sequence parameter sets (SPS). Next, the PPSs containing information related to all the slices belonging to a single frame are read. In general resending the PPS with each coded frame is not required as long as the information contained in the PPS is still valid, which is often the case. As an alternative, a set of SPSs and PPSs can be sent in an out-of-band channel before transmitting the remainder of the video and each slice header contains a reference to one of the previously transmitted PPSs. Once the parameter sets are received, the video is decoded frame by frame. As explained before, a frame comprises a sequence of slices, each slice being decodable independently. The decoding process for one slice is simple. The slice header contains the number of the first MB in the slice, which is decoded first. The remainder of the slice is then decoded MB by MB, the information of the previous MBs being used to decode their neighbors present in the same slice. The key difference when FMO is used is that the next MB to be decoded may not be in raster scan order. In this case, a few bits are allocated to the encoding of the extra information needed to find the next MB in the PPS and in the slice headers depending on the FMO Type used (see section 2.2.1).

3.1.1 Overview

The proposed scheme rearranges the slice structure to form a new slice map where all the MBs in a slice are successive MBs in raster scan order. In this way, the FMO structure is removed. A flowchart of the algorithm is presented in Figure 6. For the parameters sets, the FMO-related information is removed and the remaining information is written to the output file. When a frame arrives, the entropy decoding step extracts the information related to all received MBs (belonging to the received slices). Then a new slice map is created: the frame is scanned in

raster scan order and as long as no MB is missing, MBs are added to the same slice. If one MB is missing and some received MBs have not been processed yet, the frame is further scanned until the next received MB is found. Because the previous MB in raster scan order is missing, this MB constitutes the beginning of a new slice. The same process is applied until all received MBs belong to a slice. Once the last received MB has been found, the frame is sent to the output and the scheme is applied to the following frame. Since this process is frame-based and can be activated as soon as an entire frame has been received, it does not require the entire video to be downloaded, which makes it suitable for real-time video streaming applications. In addition, since the computations are performed in the compressed domain and the raw video is not reconstructed, storage requirements are reduced.

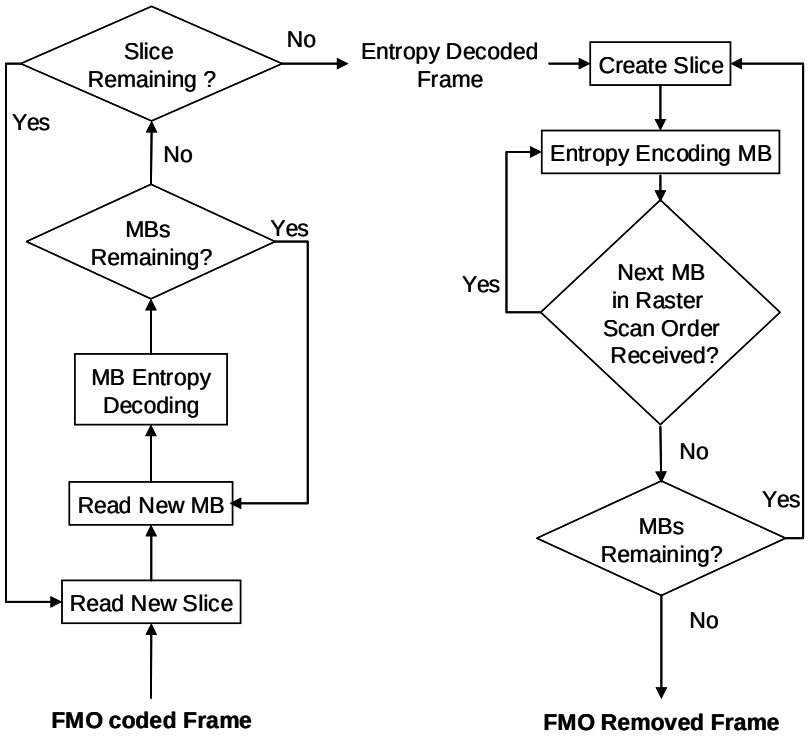
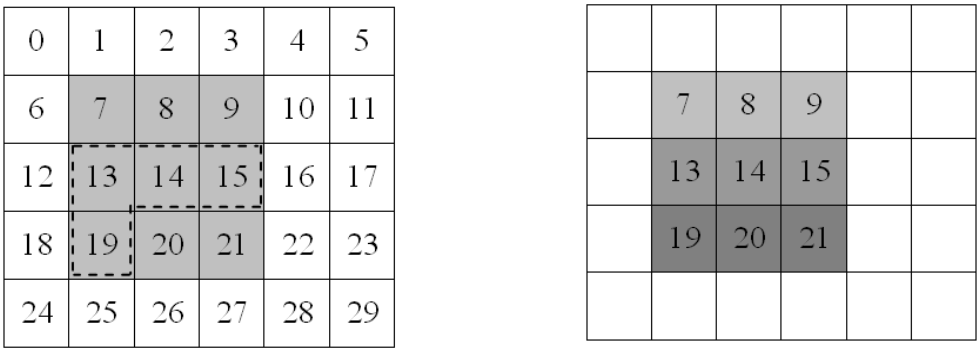


Figure 6 – FMO removal algorithm flowchart.

3.1.2 Limitations

When a new MB is coded within the same slice, the information from previously coded MBs is used to improve the coding efficiency. For inter-predicted MBs, this allows a reduction in the length of the syntax elements. Removing this dependency and coding a MB and its neighbors in different slices is always possible. On the other hand, for intra-coded MBs, neighboring MBs are used to predict textures. Because prediction is not allowed across slice boundary, assigning an intra-predicted MB to a slice and its predictors to another is impossible. For example, in Figure 7(a), MB 20 is predicted from MBs 13, 14, 15, and 19 and they all belong to the gray slice. If the white slice is lost, three new slices will be created by the proposed scheme as shown in Figure 7(b) and MBs 20 and 13, 14, 15 will belong to two different slices thus preventing the intra-prediction. Therefore, intra-predicted blocks prevent the use of the proposed scheme. As a result, the FMO removal process cannot be applied to frames containing intra-predicted blocks, including I-frames. Our implementation focuses on P-frames with intra-prediction disabled but since the reference frames remain unchanged during the FMO removal, the process can be applied in the same way for B-frames.



(a) FMO-encoded frame

(b) FMO-removed frame after loss of the white slice

Figure 7 – Example when intra-prediction is allowed. The different gray levels correspond to different slices.

3.1.3 Macroblock Treatment

During the encoding, inter-predicted MBs can be subdivided in smaller blocks. There are four partition sizes for inter-predicted MBs in H.264: 16x16, 16x8, 8x16, or 8x8 blocks. In the 8x8 partition, each 8x8 block can be further subdivided into up to four 4x4 blocks. In all cases, each block is assigned a MV pointing to an area in the reference frames used for prediction. Once the MVs have been computed, the residual information for luma and chroma components is computed. However, in order to improve the compression efficiency, all of this information is not coded directly into the bitstream. First, motion compensation is applied, that is, the horizontal and vertical components of the MVs of each block are predicted from the neighboring blocks and the difference between the actual MV and its prediction is coded into the bitstream. Next, if any luma or AC-chroma residual information is present, the number of non-zero coefficients for each 4x4 luma or AC-chroma block is predicted from the neighboring blocks. Based on the values obtained, on the actual number of non-zero coefficients, and on the number of trailing ones of the block, a codeword is chosen and coded into the bitstream. The remaining information for luma and AC-chroma blocks is then coded into the bitstream. Note that unlike the AC-chroma residuals, the DC-chroma residuals of each MB are independently decodable. When the MV predictions are exact and there is no residual for one MB, no information is coded into the bitstream except the fact that this MB is skipped.

We now describe the MB treatment in the FMO removal scheme (Figure 8). During the entropy decoding step, the MVs, the number of non zero-coefficients, and the number of trailing ones are computed and stored for all MBs. Then the new slice map is created. Figure 9 shows that with the new slice map, some MBs can use new neighbors for prediction purposes: it is necessary to re-encode each MB into the new bitstream according to the new slice map. For skipped MBs in the FMO-encoded video, their new predicted MVs under the new slice map are compared to their actual motion data and, if they remain the same, the MB is skipped in the FMO-removed video,

otherwise the new MV compensations are coded. New MBs may also be skipped under the new slice map: each MB without residual information and predicted with a 16x16 partition is tested and, if the new MV predictions correspond to its actual motion data, the MB is coded as a skipped MB.

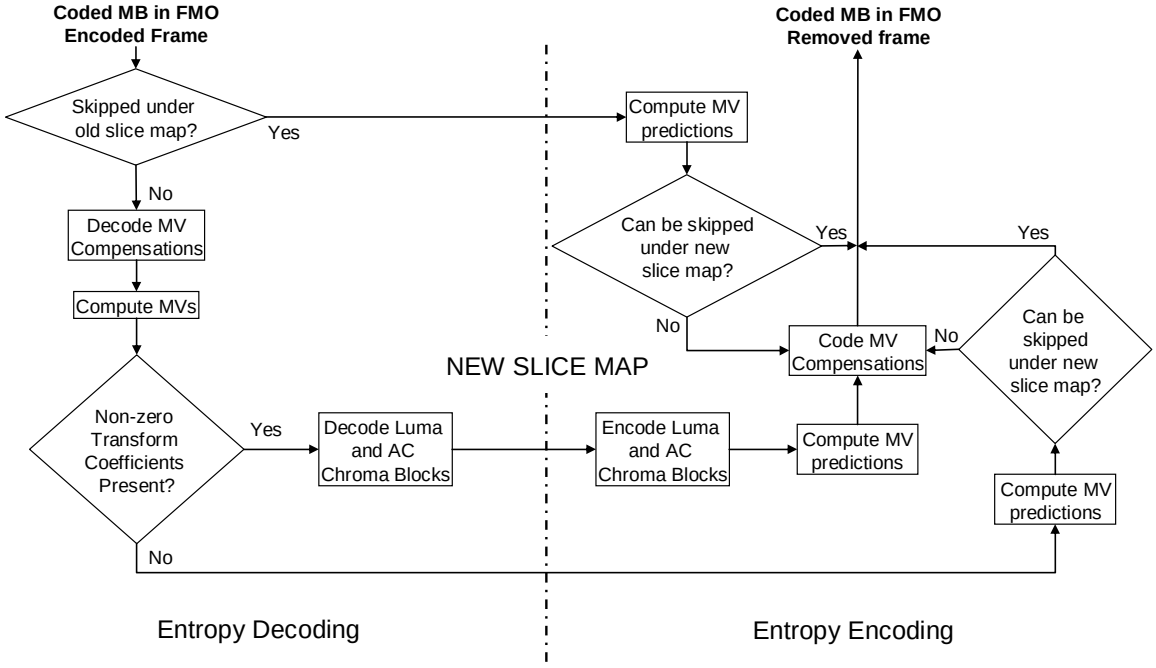


Figure 8 – MB treatment flowchart.

0	1	2	3	4	5
6	7	8	9	10	11
12	13	14	15	16	17
18	19	20	21	22	23
24	25	26	27	28	29

(a) FMO-encoded frame

0	1	2	3	4	5
6	7	8	9	10	11
12	13	14	15	16	17
18	19	20	21	22	23
24	25	26	27	28	29

(b) FMO-removed frame

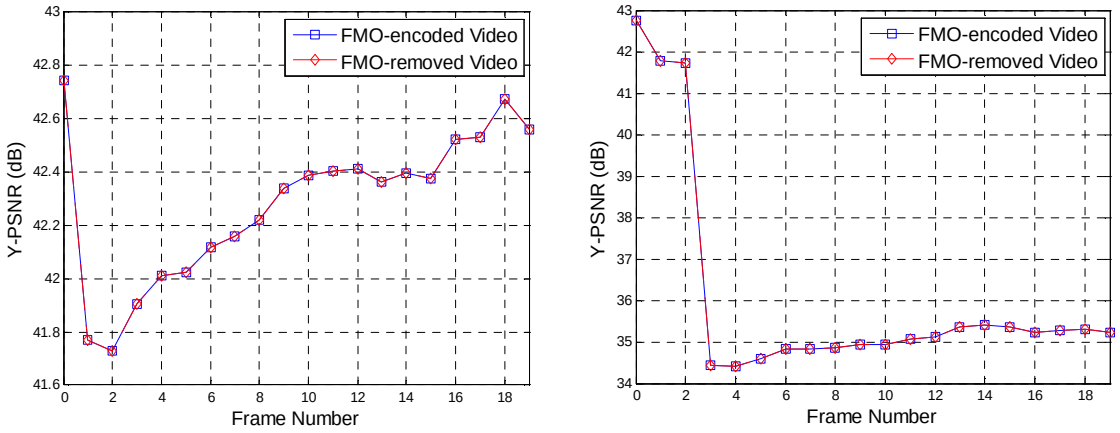
Figure 9 – Example when interlaced FMO is used.

3.2 Visual Quality Evaluation

In this section, we use the Foreman sequence in QCIF resolution (176x144 pixels per frame) encoded with a QP value of 20. The first frame is an I-frame encoded without FMO and the remainder of the video is composed of P-frames encoded with FMO Type 1 (two SGs) with one slice per SG. The experiment was carried out on other videos with different resolutions and similar results were obtained.

Our main concern when designing the proposed scheme is to keep the same visual quality between the FMO-encoded video and the FMO-removed video. In the previous section, we explained the process of removing the FMO structure from a received video. No information is lost during the processing of the algorithm. Hence, when there are no losses, the algorithm does not compromise visual quality, as demonstrated in Figure 10(a).

In a lossy environment, an additional error concealment step is needed at the decoder. However, applying the FMO removal process before the decoding does not affect its efficiency as shown in Figure 10(b). When a frame is received, all received slices are decoded before the missing MBs are concealed. When the proposed scheme is applied, the status of the frame before error concealment is the same as if FMO had not been removed. Therefore, the scheme will not affect the error concealment.



(a) No Loss occurred (b) One slice lost on frame 3

Figure 10 – Y-PSNR for the 20 first frames of the Foreman QCIF sequence.

We compare our results to the performance obtained by decoding the FMO-encoded video entirely and then re-encoding it without FMO. The setup of the experiment is presented in Figure 11. For the re-encoding, a QP value of 0 is chosen, which is the finest possible quantization in H.264. Although this process has a higher computational cost and is done at the expense of the compression efficiency, Table 4 indicates that the average Y-PSNRs of the reconstructed videos are slightly smaller. This shows that, contrary to re-encoding techniques, the FMO-removal scheme does not compromise the Y-PSNR.

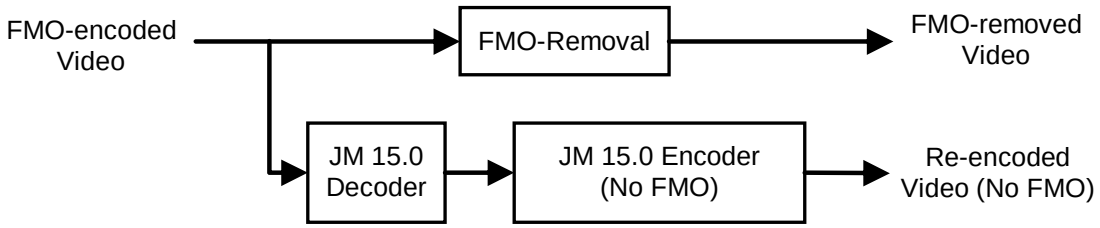


Figure 11 – Experimental setup to compare FMO-removal and re-encoding.

Table 4 – Comparison between FMO-removed videos and re-encoded videos.

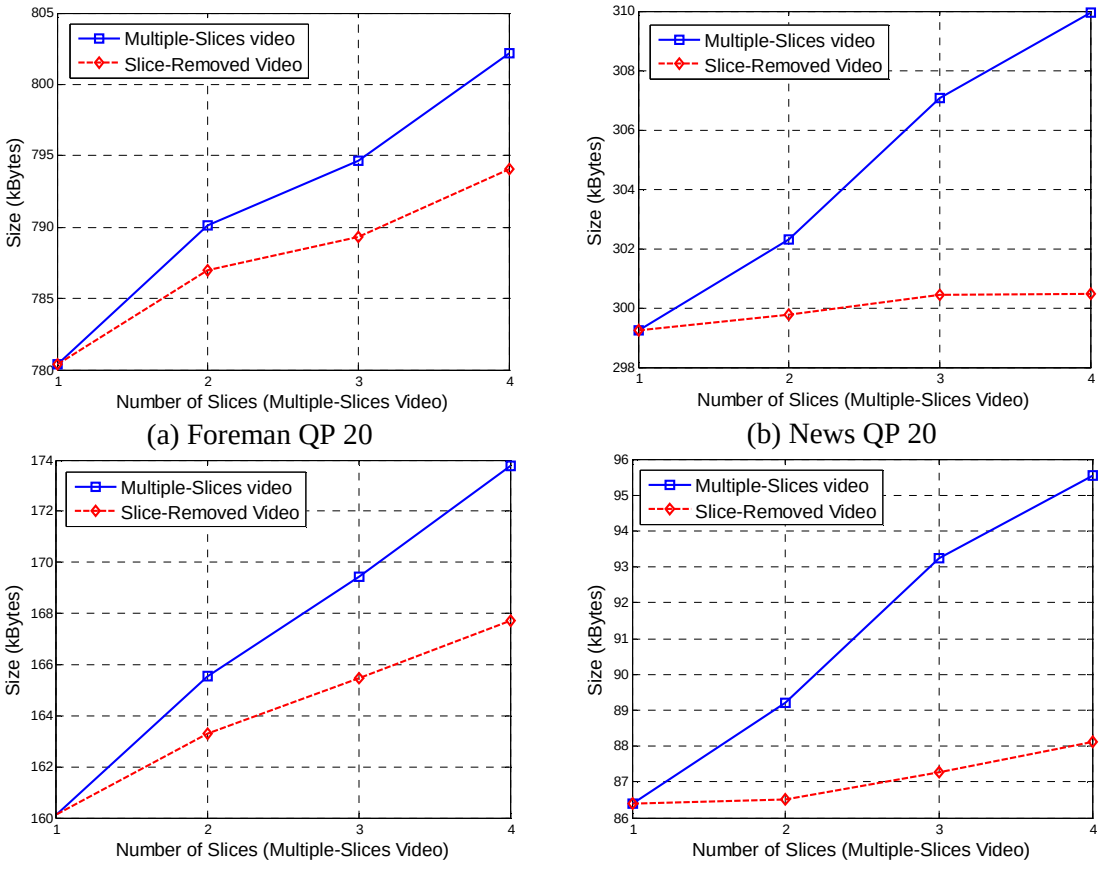
	Video	Size (bytes)	Average Y-PSNR (dB)
No Loss	FMO removed	812,683	42.7313
	Re-encoded	2,951,161	42.7241
5 % random losses	FMO removed	792,388	28.6476
	Re-encoded	2,713,557	28.6469

3.3 Using Multiple Slices

In section 3.1, we described how the proposed scheme can be used to remove the FMO structure. In this section we focus on videos encoded without FMO but with several slices. Some encoders do not have the capability of encoding with FMO. Nevertheless, the slice map can still be modified via the number of slices. For example, in Internet transport, [3] recommends encoding videos with slices of less than 1500 bytes to avoid fragmenting the packets containing

the slices. However, using several slices per frame induces overheads. Assuming a video has been encoded with several slices to improve transmission reliability, our scheme allows modification of the slice structure by grouping all the MBs in one unique slice. This reduces the size of the video.

To assess the possible gain, an experiment is carried out. The first 300 frames of the Foreman and News QCIF videos are encoded with varying numbers of slices per frame and with different QP values (20 and 30). Next, the proposed scheme is applied to each encoded video so that only one slice per frame remains at the output (no losses occurred). Figure 12 presents the sizes of the different files before and after applying the method. For one slice per frame, the slice structures are the same and the two files are identical. As expected, with several slices per frame, the output video is always smaller but the savings vary with the videos and the QP values.



(a) Foreman QP 20 (b) News QP 20

(c) Foreman QP 30 (d) News QP 30

Figure 12 – Sizes of the videos after FMO removal when varying the number of slices.

The reduction in file size does not only depend on removing slice-related information but is also due to improvements in coding efficiency, especially for the choice of the codewords for luma and chroma residual information. This is the reason why the gains are higher for lower QP values. Next, the gains increase with the number of slices because more MBs have less neighbors and more slice-related information can be removed when more slices were used.

Finally, removing the slice structure does not allow recovery of the size of the video directly encoded with one slice. This is due to the dependency of the MB partitioning on the slice structure (see section 3.4.1 for more details).

3.4 Overheads

In section 3.3, we explained how the proposed scheme may reduce the size of the received videos. In a lossy environment, the loss of a slice may result in the creation of new slices requiring extra bits to code the headers. This is not the only reason for a change in the video size: in some cases, breaking the slice structure may degrade the efficiency of entropy coding.

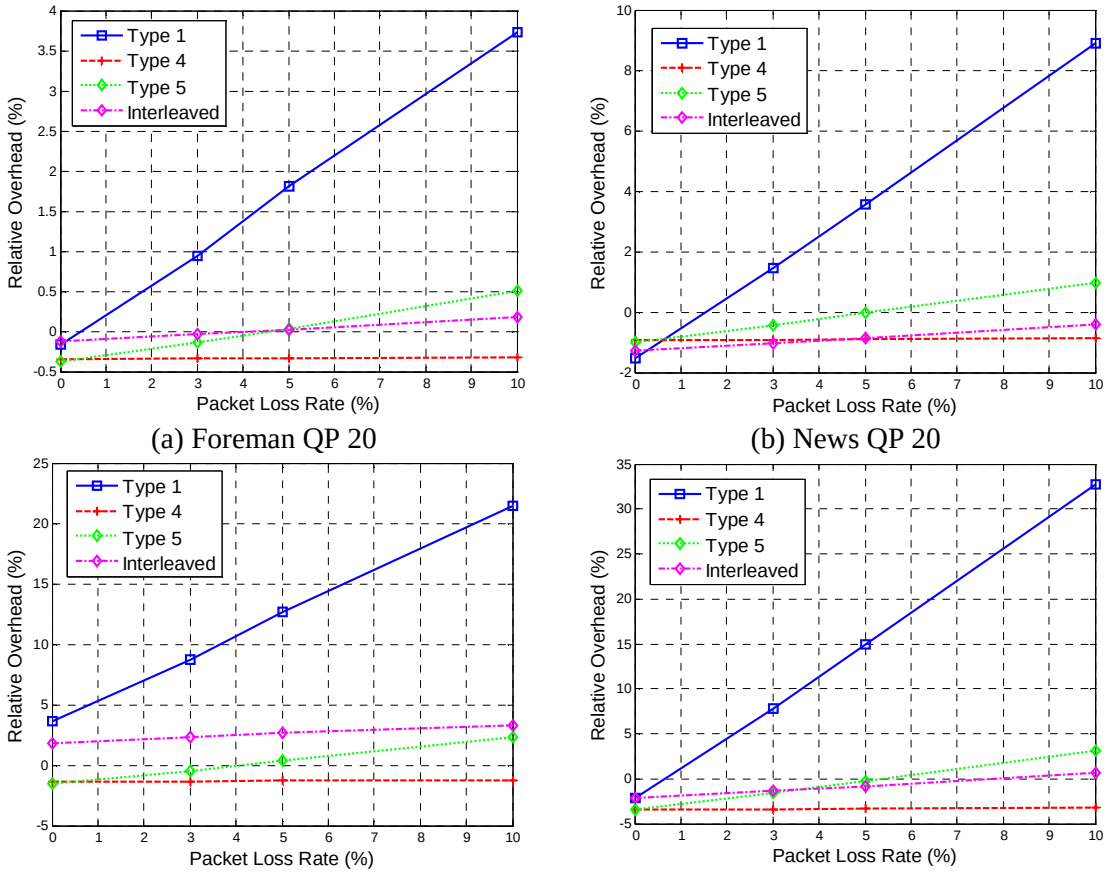
In this section, we assess the relative overheads incurred by our scheme for various FMO Types and we propose a model to predict the overhead for FMO Type 1. Although the presented experiments are performed on a fixed FMO slice mapping, the scheme can apply to videos where the slice mapping changes with each frame. The implementation of the method has been done by modifying the JM 15.0 reference software.

3.4.1 Experiments

We consider the transport of videos over a lossy network subject to uniform random losses. I-slices and parameter sets are assumed never to be lost. We vary the packet loss rate between 0% and 10% and compute the relative overheads between the FMO-encoded video and the corresponding FMO-removed video. The relative overhead *r* is defined as:

$$r = \frac{[size(FMO\_removed\_Video) - size(FMO\_encoded\_Video)]}{size(FMO\_encoded\_Video)} \quad (1)$$

The experiments are carried out on the same videos as in section 3.3. The first frame of each sequence is an I-frame encoded without FMO and all other frames are FMO-encoded P-frames. In all cases, each packet transports only one slice. Figure 13 presents the average results over 10 traces.



(a) Foreman QP 20 (b) News QP 20

(c) Foreman QP 30 (d) News QP 30

Figure 13 – Relative overheads after FMO removal for different packet loss rates.

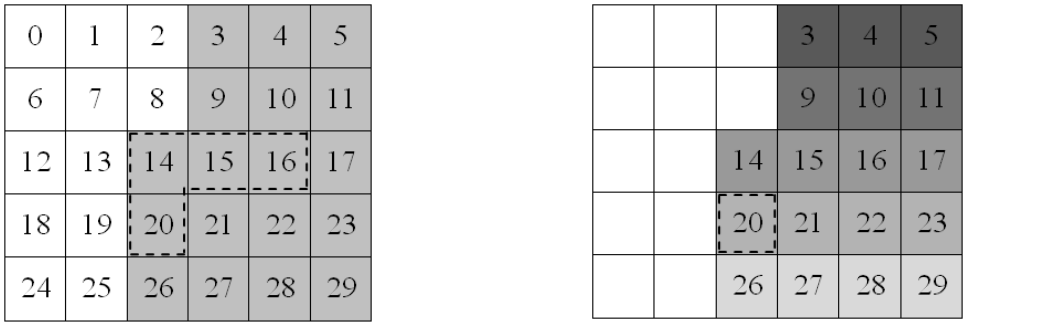
As expected, the relative overheads strongly depend on the original slice map. When there are no losses, only one slice per frame is required in the FMO-removed frames regardless of the original slice map. When the proposed scheme is applied after slice losses, the slice map in the FMO-removed frames depends on the original slice structure.

Even in loss free environment (0% packet loss), the relative overheads differ and are difficult to predict. In most cases, they remain negative but they can turn positive as shown in Figure 13(c) for FMO Type 1 and Interleaved. In this particular case, the bits saved by removing slice-related information are overcome by the loss in coding efficiency. This is due to the process of selection of the prediction mode at the encoder [7]. MB partition modes are chosen to minimize the number of bits required to code the MV differences. Because the MV predictors depend on the MVs of the available neighbors, the selected partition mode depends on the slice map used by the encoder. After the FMO-removal process, the slice map is changed and the MB partition modes may not be optimal anymore: this may result in a positive overhead. This phenomenon may happen with any slice map but it is significant only when the number of neighbors used for prediction is reduced, for example, for FMO Type 1 and Interleaved, where no more than one neighboring MB can be used for MV prediction.

On the other hand, the number of non-zero coefficients and trailing ones do not depend on the partition mode: the higher the number of neighbors, the smaller the codewords. When an entire frame is correctly received, the FMO-removed frame has only one slice and the number of available neighbors is maximized. By decreasing the QP value, the number of bits required to code the MV differences remains the same (the prediction does not depend on the QP value) but the number of residual coefficients increases. This increase results in more coded codewords or in longer codewords, both effects inducing higher savings when increasing the number of neighbors. When the QP value is sufficiently small the savings on codewords overcome the possible worsening of the MV prediction under the new slice map. This is why all overheads for 0% loss turn negative for a QP value of 20 in our experiment.

Another observation is the increase in relative overhead when the packet loss rate increases. For FMO Type 1, 5, and Interleaved, this is due to the increase in the number of slices as shown in Figure 14. This results in an increase in the relative overhead for two reasons. First, Figure 14 shows that creating new slices may result in reducing the number of neighbors for some MBs and

thus, the coding efficiency. Second, each new slice requires a new slice header and additional bits (see section 3.4.2 for more details). This overhead is almost constant and does not depend on the QP value. This explains why the relative overheads observed for low QP values remain smaller: the absolute values of the overheads are of the same order but the size of the videos with low QP values is bigger. Thus, the relative overhead is higher. The same applies for the News video which shows little motion and details and thus is small compared to the Foreman video (306,601 bytes versus 814,070 bytes for QP 20 and FMO Type 1): its relative overheads are higher.



(a) FMO-encoded frame with FMO Type 5

(b) FMO-removed frame after loss of the white slice

Figure 14 – Reduction of the number of neighboring MBs after loss and FMO removal.

3.4.2 Overhead Prediction for FMO Type 1

The use of FMO Type 1 provides the best error resilience but also results in the lowest compression efficiency (section 2.2.2). When applying our scheme at the receiver, the increase in relative overhead becomes even more dramatic (Figure 13). This is expected since, for FMO Type 1, when a slice is lost, consecutive MBs in raster scan order are not present in the frame. As a result, for each received MB, a new slice has to be created and no neighboring MBs can be used for improving the coding efficiency. This is the worst possible situation. Having a model for the overhead allows predicting the storage space for the receiving device which can be useful in applications for which the video has to be stored before transmission such as video on demand.

The general expression for the overhead is:

$$a = \sum_i (\tilde{f}_i - f_i) \tag{2}$$

where $\tilde{f}_i$ is the size of the i^{th} frame of the FMO-removed video, and f_i is the size of the i^{th} frame of the FMO-encoded video. Taking the expected value on both sides:

$$E[a] = E\left[\sum_i (\tilde{f}_i - f_i)\right] = \sum_i E\left[(\tilde{f}_i - f_i)\right] \tag{3}$$

Assuming that the packet loss rate probability p is known, we derive the following expression:

$$E\left[(\tilde{f}_i - f_i)\right] = p(1-p)S1_i + p(1-p)S2_i + (1-p)^2 a_i^0 \tag{4}$$

where $S1_i$ and $S2_i$ are the overheads induced by the loss of one slice when the other is received and a_i^0 is the overhead when both slices are received. If the degradation of coding efficiency due to the unavailability of neighbors is negligible compared to the size of the new headers and other slice-related extra bits, we can assume that $S1_i$ and $S2_i$ do not depend on the frame content. In other words, $S1_i$ and $S2_i$ are almost the same for all frames and we can drop the subscripts. Equation (3) becomes:

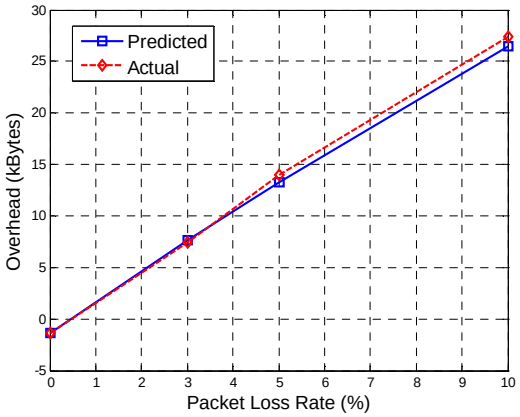
$$\begin{aligned} E[a] &\approx np(1-p)(S1+S2) + (1-p)^2 \sum_i a_i^0 \\ E[a] &\approx np(1-p)(S1+S2) + (1-p)^2 a^0 \end{aligned} \tag{5}$$

where n is the number of frames from which FMO Type 1 has been removed and a^0 is the overhead for the entire video when no loss has occurred. a^0 is known before transmitting the video. $S1$ and $S2$ can be predicted as follows. In one frame, all new slice headers are the same except for the bits coding the number of the first MB of the slice. However, because the slice structure is predetermined with FMO Type 1, when one slice is lost the location of the remaining MBs is known: we can predict exactly the number of additional bits. Next, a 3-byte flag is added before each new slice for parsing at the decoder. Finally, we have to take into account the fact that each slice must be coded in an integer number of bytes. When the information bits do not

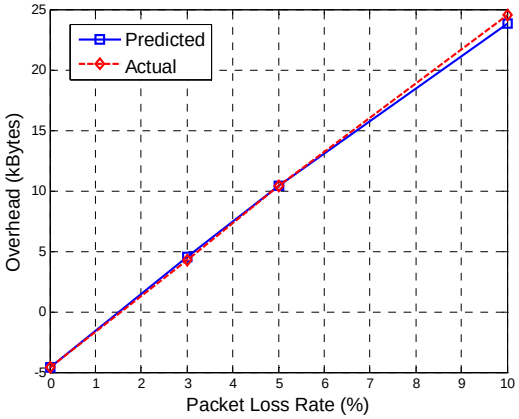
sum up to an integer number of bytes, some stuffing bits are added at the end of the slice. With two slices per frame, these extra bits produce a negligible overhead. However, this no longer holds when the number of slices is half the number of MBs per frame which is the case when one of the two slices is lost for FMO Type 1. We model the number of stuffing bits as a uniform random variable: one byte is added with a probability of 0.5 for each new slice. Table 5 and Figure 15 depict the accuracy of the prediction model. As can be seen from the experiments, the model is in general very accurate, with a relative error less than 6% for packets loss rates up to 10%.

Table 5 – Average actual and predicted overheads after FMO removal for FMO Type 1.

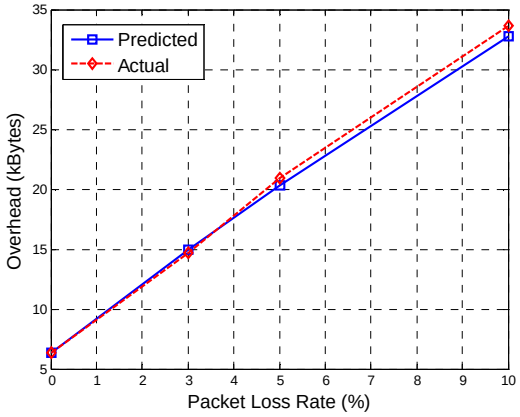
Video	**Foreman QCIF**		**News QCIF**	
QP	20	30	20	30
		***p* = 3%**		
Predicted (bytes)	7606	14918	4542	7092
Actual (bytes)	7445	14734	4295	6894
Relative error	2.17%	1.25%	5.75%	2.87%
		***p* = 5%**		
Predicted (bytes)	13294	20308	10355	12801
Actual (bytes)	13981	20947	10391	12909
Relative error	4.91%	3.05%	0.35%	0.84%
		***p* =10%**		
Predicted (bytes)	26437	32732	23799	25994
Actual (bytes)	27383	33640	24545	26882
Relative error	3.45%	2.70%	3.04%	3.30%



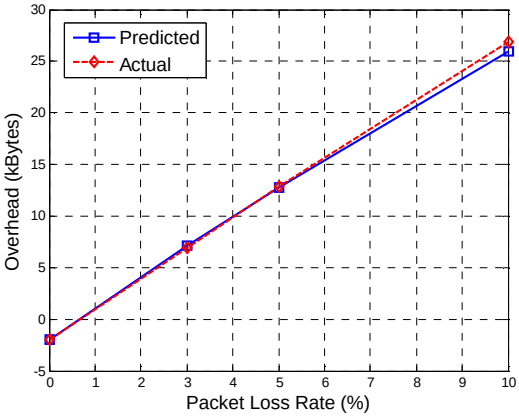
(a) Foreman QP 20



(b) News QP 20



(c) Foreman QP 30



(d) News QP 30

Figure 15 – Actual average overheads after FMO removal and their predicted values for FMO Type 1 under different packet loss rates.

CHAPTER 4

TEMPORAL ERROR CONCEALMENT FOR I-FRAMES

The FMO removal scheme presented in Chapter 3 enables efficient error concealment for P- and B-frames on all H.264 decoders thanks to the use of FMO. However, we explained in section 3.1.2 that intra-predicted MBs prevent the use of the technique and thus FMO cannot be used to protect I-frames when the decoder does not support the Baseline or Extended profiles. This justifies the need for an efficient error concealment algorithm for I-frames even when FMO is not employed.

In this chapter we introduce a new error concealment scheme specifically designed for protection of I-frames. This new scheme can be integrated with the FMO-removal method proposed previously, which only deals with P-frames or B-frames. In section 4.1 we present an overview of the currently used error concealment techniques for I-frames before introducing our algorithm in section 4.2. Eventually, we compare the performances of the proposed methods to some other algorithms in section 4.3.

4.1 Error Concealment for H.264

Error concealment is not a standardized feature of H.264. Nevertheless, the H.264 JM reference software provides error concealment algorithms for partial and entire frame losses. We only focus on partial frame loss, which is typically the case when multiple slices are employed.

In the case of P-frames, motion compensated error concealment is performed to repair the damaged part of the frame. Assuming that the motion is smooth and continuous in a frame, the MVs of the lost MBs are predicted from the received surrounding MBs. For I-frames, the JM error concealment algorithm uses a weighted sample average of the pixel values of the

surrounding MBs [7] (Figure 16). The authors in [8] and [9] refine this algorithm by using the deblocking filter and data hiding respectively. These spatial approaches conceal the missing blocks without using other frames and are justified when the I-frames do not resemble the preceding frames (e.g., in a scene change). However, they give poor results for high error rates and are generally not efficient for concealing large areas of the frame.

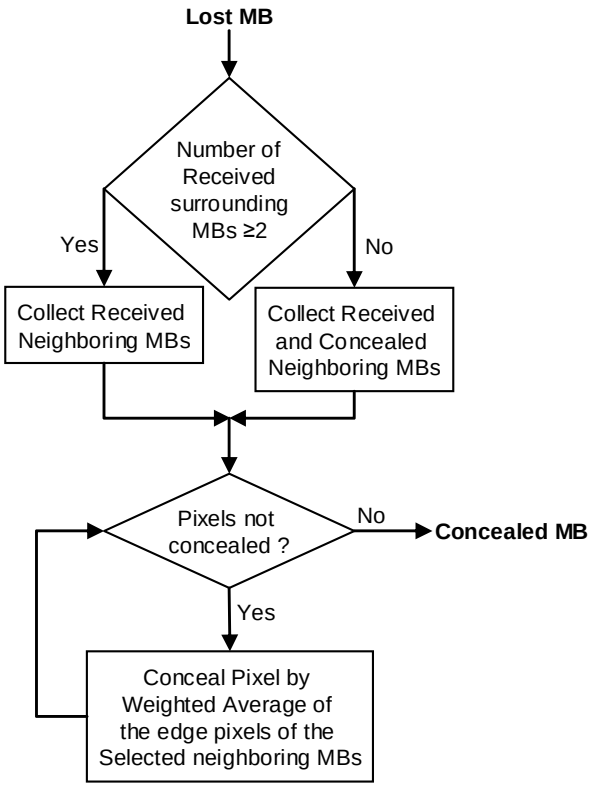


Figure 16 – Error concealment for I-frames in the JM reference software.

In error-prone environments, I-frames are used even without scene changes to prevent error propagation. In this case, the content of the I-frame may be highly correlated with the previous frame. This implies that the use of temporal error concealment may be more efficient than spatial techniques. However, because I-frames are entirely intra-predicted, the bitstream does not directly provide the motion data needed to perform error concealment. In [10], a strategy is presented for predicting possible MVs for the missing MBs. Unlike the previous temporal error concealment

methods, our algorithm works on 8x8 blocks and optimizes the MV search by using the motion data of the previous frame.

4.2 Algorithm Details

The concealment is performed once all the received MBs of the frame have been decoded. First, a status map of the MBs is created. One MB can be correctly received, concealed or lost. When a MB is concealed, the status map is updated. The concealment is done column-wise. Because the center of the frame is usually more difficult to conceal, the left-most and right-most corrupted columns are concealed successively until all columns have been repaired. To conceal a column, the same process is applied with the upper and lower corrupted MBs. Figure 17 presents a possible status map during error concealment.

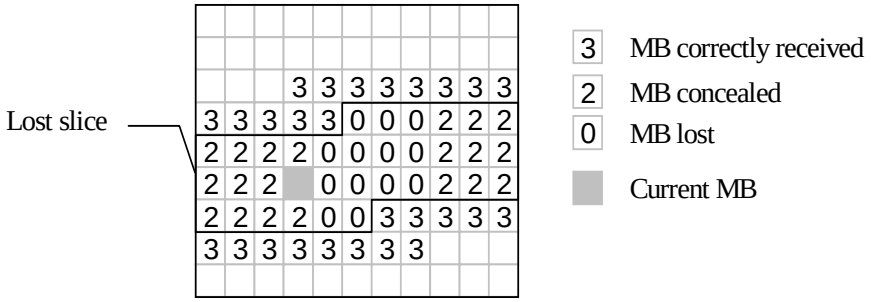
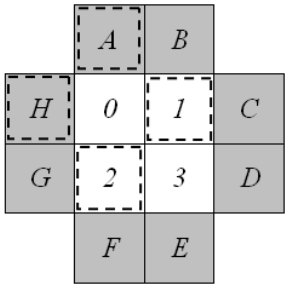
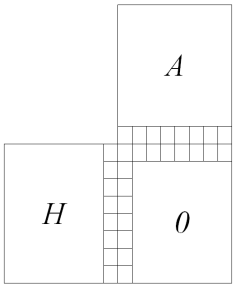


Figure 17 – Possible status map during concealment.



(a) Blocks *0* to *3* belong to the lost MB



(b) Luma samples used for edge distortion considerations

Figure 18 – Subdivision of the MBs for concealment. For block *0*, blocks *A*, *H*, *1* and *2* may be used for concealment depending on their availability.

The 16x16 MB to be concealed and its surrounding MBs are subdivided into 8x8 blocks. In Figure 18(a), the blocks *0* to *3* correspond to the lost MB and the gray blocks correspond to neighbouring MBs. A flowchart of the concealment process for a MB is presented in Figure 19.

First, we check the status of the surrounding blocks from other MBs. For block *0*, this corresponds to blocks *A* and *H*. If both have been correctly received, the two corresponding MVs are computed. If only one of them has been correctly received, only this block is used. Otherwise, if at least one of them has been concealed, the same treatment as if they had been correctly received is applied. Otherwise, other 8x8 blocks from the same MB have already been concealed and they are used the same way. Note that when blocks have been concealed their MVs have already been computed.

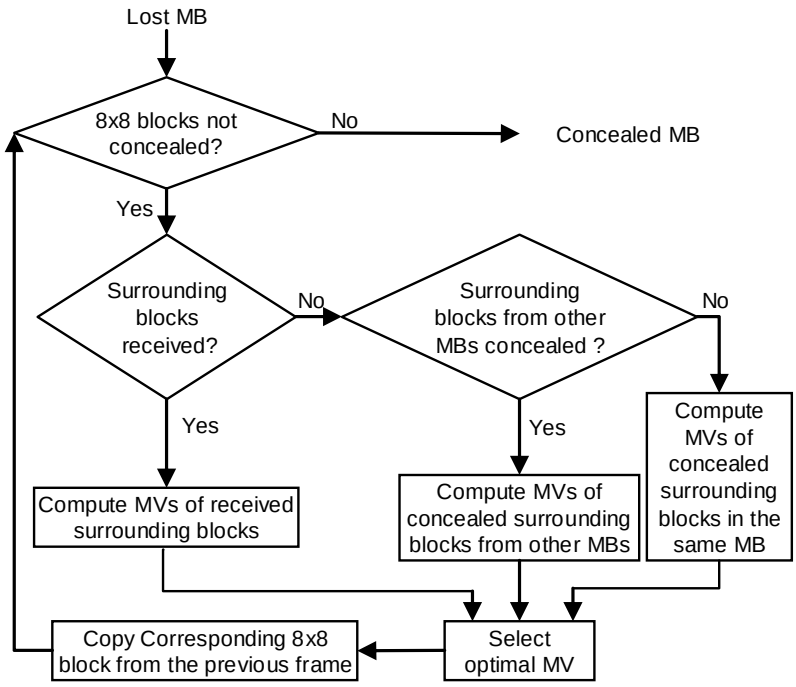


Figure 19 – Error concealment algorithm flowchart.

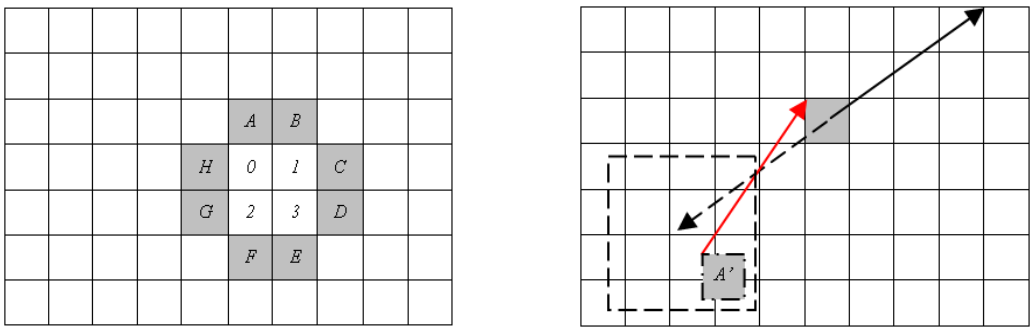
The MV of an 8x8 block is computed by searching a matching area in the previous frame (Figure 20). Suppose the MV of block *A* must be computed. If the motion is smooth and continuous across the frames, the matching block is probably close to the location of *A* displaced

by the MV of the block at the same location in the previous frame. If this MV is not available, it is considered null. Next, the search is performed on a small area around this location and the correct block *A'* is the block minimizing ε_A with

$$\varepsilon_A = \sum_i (A[i] - C[i])^2 \quad (6)$$

where $X[i]$ is the i^{th} luma component of block X and C is an 8x8 candidate block. Eventually, the MV for *A* corresponds to the translation from *A'* to *A*.

Once the different MVs are obtained, the correct MV for error concealment is determined by trial. If two MVs have been computed in the previous step, four candidate MVs are considered: the null MV, each previously computed MVs, and the average of the two MVs. If only one MV has been computed, then it is compared to the null MV. The correct MV minimizes the edge distortion with the received or already concealed surrounding blocks. This distortion is measured as the sum of the absolute luma sample value differences on the edges of the blocks. Figure 18(b) presents an example of the considered pixels when blocks *A* and *H* have been received or concealed and block *0* is being concealed. This process of electing the concealing MV reduces the production of visible blocky effects.



(a) frame N (b) frame N-1

Figure 20 – MV search process for one 8x8 intra-predicted block. The black solid arrow is the MV of the block at the same location as *A* in frame N-1 and the red arrow is the computed MV.

4.3 Error Concealment Performances

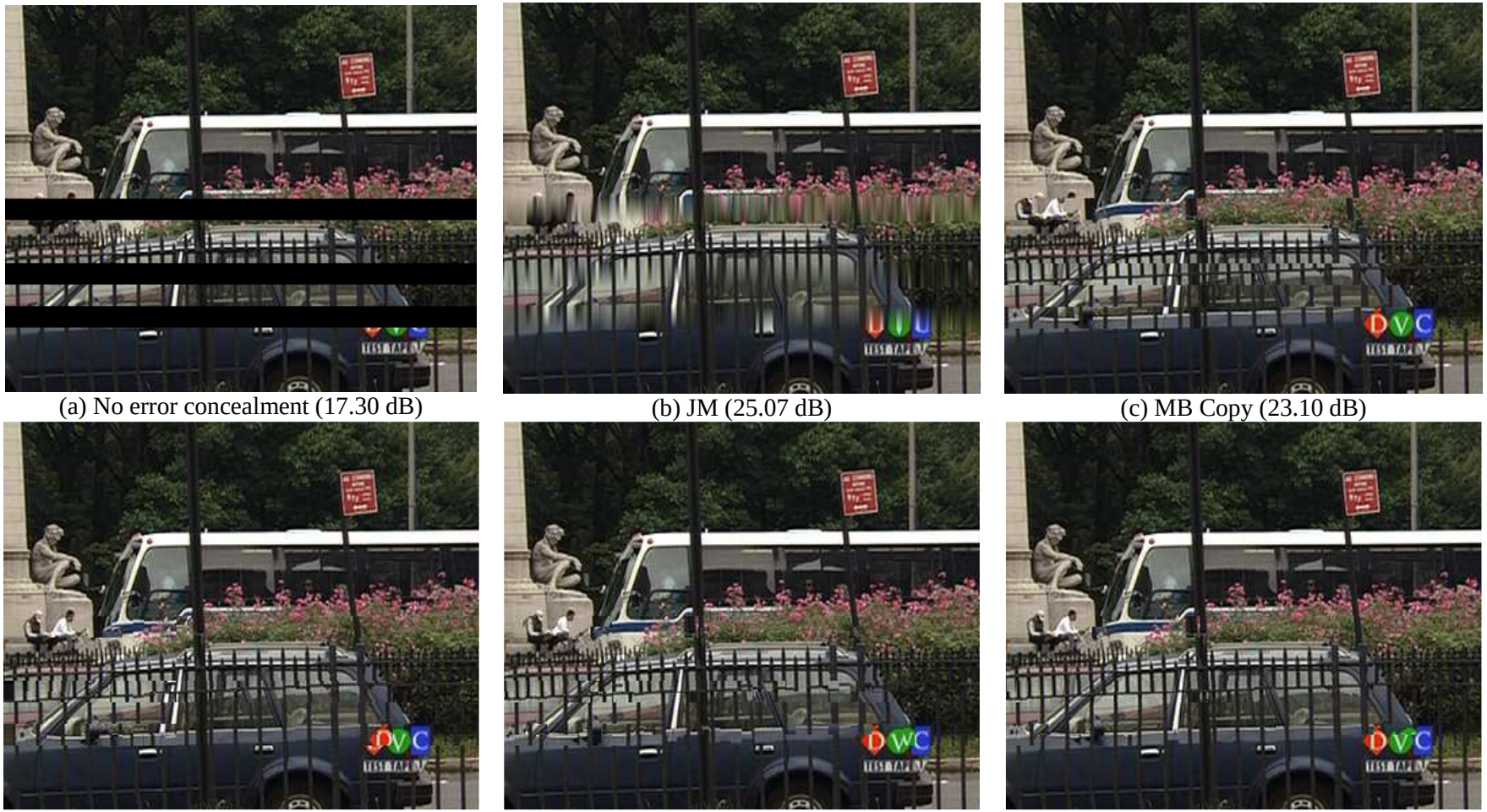
In this section, we present the results obtained for the Foreman and Bus CIF videos and for the Shields and Parkrun HD videos (1280x720 pixels). All videos are encoded with a QP value of 20. Note that the Bus and Parkrun videos are faster moving. Each video is encoded with and without FMO (Type 1 with two SGs) with slices of 22 MBs for the CIF videos and slices of 160 MBs for the HD videos. An I-frame is introduced for every 10 frames and the other frames are P-frames. There is no scene change, thus this corresponds to a situation in which I-frames are used for preventing loss propagation. I-frames are subject to uniform random packet loss with probabilities of 10% and 20%. The P-frames are not corrupted but are affected by the errors propagating from the degraded I-frames which are used as references. The proposed method is compared to the JM 15.0 algorithm, to the MB copy algorithm (i.e., if a MB is lost, it is replaced by the MB at the same location in the previous frame), and to the MV copy algorithm (if a block is lost, the MV corresponding to the same block in the previous frame is used). A less computationally intensive version of our algorithm (Reduced) is also presented. In this case, instead of comparing all luma samples of the 8x8 blocks in (6), only the samples at the edges of the blocks are compared. This reduces the computation by more than 50%. Because the proposed algorithm requires previous frames, the first I-frame of each sequence is concealed with the JM 15.0 error concealment. The results are presented in Table 6 and in Figure 21 and Figure 22.

The proposed algorithm clearly outperforms the other methods in all situations. In terms of Y-PSNR, it achieves significantly better results of several dB. Moreover, as can be seen from Figure 21 and Figure 22, the visual artifacts are also reduced and the subjective visual quality is improved. The best performance is obtained for FMO-encoded videos but the results for non FMO-encoded video are also acceptable. The simplified algorithm performs better than other techniques for most videos and in all cases when FMO is used. Note that the MB and MV Copy performances depend on the motion content. On the Bus video (fast moving), the MV Copy

algorithm outperforms the MB Copy while it is the opposite on the Foreman video. However, in both cases, the proposed method gives better results. As expected, the JM error concealment performs better when FMO is used since smaller areas need to be concealed which makes the interpolation more efficient.

Table 6 – Average Y-PSNR after error concealment in dB.

	Bus (150 frames)		Foreman (300 frames)		Parkrun (100 frames)		Shields (100 frames)	
Method	**No FMO**	**FMO**	**No FMO**	**FMO**	**No FMO**	**FMO**	**No FMO**	**FMO**
				10% loss				
JM	26.45	30.21	29.26	31.55	27.48	29.45	30.31	33.06
MB Copy	28.33	29.00	33.01	35.30	29.93	29.08	33.54	32.99
MV Copy	29.27	29.93	31.80	33.55	31.28	30.08	34.58	34.36
Reduced	29.76	31.95	33.86	36.01	30.41	30.23	35.17	36.47
Proposed	32.87	34.28	34.54	37.09	31.75	32.77	36.63	38.55
				20% loss				
JM	24.40	24.61	26.71	27.93	24.17	26.78	27.50	31.10
MB Copy	25.46	25.49	31.00	30.41	26.49	25.89	30.35	30.14
MV Copy	26.27	26.24	30.87	29.70	27.71	27.19	31.55	31.68
Reduced	26.77	27.50	30.94	31.64	27.01	27.21	31.93	34.33
Proposed	29.79	29.86	31.74	33.20	28.49	30.25	33.46	37.18



(a) No error concealment (17.30 dB) (b) JM (25.07 dB) (c) MB Copy (23.10 dB)

(d) MV copy (23.99 dB) (e) Reduced (26.07 dB) (f) Proposed (29.96 dB)

Figure 21 – Frame 10 of the non-FMO Bus Video with corresponding Y-PSNR. Observe the fence in the foreground.

(a) No error concealment (11.61 dB) (b) JM (25.60 dB) (c) MB Copy (32.01 dB)

(d) MV Copy (29.65 dB) (e) Reduced (33.41 dB) (f) Proposed (33.93 dB)

Figure 22 – Frame 60 of the FMO Foreman video with corresponding Y-PSNR.

CHAPTER 5

CONCLUSION

In this thesis, we presented a new lossless FMO removal scheme to remove the FMO structure from non FMO-encoded videos. This is done after evaluating the usefulness of FMO and its overheads. Extending the FMO removal scheme to non FMO-encoded videos with multiple slices per frame can help reducing their size. The proposed method works in the compressed domain and does not require extended computations or memory. Moreover it is frame based which makes it suitable for live video streaming applications. Our experiments show that it may induce some overheads, especially in lossy environments. Thus, we propose a model for predicting those overheads when FMO Type 1 is employed.

We also described a complementary error concealment method for I-frames that is shown to be superior to the method adopted in the JM reference software and to other classical algorithms. This technique is efficient even when MBs in raster scan order are lost and can therefore be applied without FMO. Thus, this new scheme can be integrated with the FMO-removal scheme proposed previously, which can only deal with P- and B-frames.

REFERENCES

[1] *Advanced Video Coding for Audiovisual Services*, ITU-T Rec. H.264 | ISO/IEC 14496-10 version 8, 2007.

[2] P. Lambert, W. De Neve, Y. Dhondt, R. Van de Walle, "Flexible macroblock ordering in H.264/AVC", *Elsevier Journal of Visual Communications and Image Representation*, 2006.

[3] S. Wenger, "H.264/AVC Over IP", *IEEE Transactions on Circuits and Systems for Video Technology*, July 2003.

[4] D. Marpe, H. Schwarz, T. Wiegand, **"**Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard**",** *IEEE Transactions on Circuits and Systems for Video Technology*, July 2003.

[5] M. Naccari, G. Bressan, M. Tagliasacchi, F. Pereira, S. Turbaro, "Unequal Error Protection Based On Flexible Macroblock Ordering for Robust H.264/AVC Video Transcoding", *Picture Coding Symposium*, November 2007.

[6] W. Tan, E. Setton, J. Apostolopoulos, "Lossless FMO and Slice Structure Modification for Compressed H.264 Video", *IEEE International Conference on Image Processing*, December 2007.

[7] "Joint Model Reference Encoding Methods and Decoding Concealment Methods", JVT-I049d0, San Diego, CA, USA, September 2003.

[8] W-P. Lee, T-M. Liu, C-Y. Lee, “A Joint Architecture of Error-Concealed Deblocking Filter for H.264/AVC Video Transmission”, *Internal Symposium on VLSI Design, Automation and Test*, April 2007.

[9] A. Yilmaz, A. Aydin Alatan, “Error Concealment of Video Sequences by Data Hiding”, *IEEE International Conference on Image Processing*, December 2003.

[10] P. Nasiopoulos, L. Coria-Mendoza, H. Mansour, A. Golikeri, “An Improved Error Concealment Algorithm for Intra-frames in H.264/AVC”, *IEEE International Symposium on Circuits and Systems*, May 2005.

VITA

Camille Mazataud was born in Toul, France, in 1986. He completed his Diplome d'Ingenieur Degree, from Supelec, Paris, France. He is currently working towards a Masters degree (with Thesis option) in the School of Electrical and Computer Engineering at the Georgia Institute of Technology. His research interests include networked video transport and error containment in video coding.

List of Publications:

- C. Mazataud and B. Bing, "A Practical Survey of H.264", *IEEE/ACM CNSR Conference*, Moncton (Canada), May 2009.
- C. Mazataud and B. Bing, "A New Lossless FMO Removal Scheme for H.264 Videos", *IEEE ICCCN Conference*, San Francisco (USA), August 2009.
- C. Mazataud and B. Bing, "New Error Containment Schemes for H.264 Decoders", *IEEE GLOBECOM Conference*, Hawaii (USA), December 2009.

Submitted Publications:

- C. Mazataud and B. Bing, "Macroblock Restructuring and Error Concealment for H.264 Decoders", *IEEE Transactions on Broadcasting*.